7a

N91-21942

Engineering Software Development with HyperCard

Robert J. Darko

Systems Programmer/Analyst

Hudson Products Corporation

A case study describing the successful and unsuccessful techniques used in the development of software using HyperCard. An evaluation of the viability of HyperCard for engineering and a discussion of the future use of HyperCard by this particular group of developers.

Engineering Software Development with HyperCard

Abstract:
A case study describing the successful and unsuccessful techniques used in the development of software using HyperCard. An evaluation of the viability of HyperCard for engineering and a discussion of the future use of HyperCard by this particular group of developers.

Introduction:

A brief description of Hudson Products Corporation is necessary to keep the rest of this paper in perspective. Hudson Products Corporation's major products are Air Cooled Heat Exchangers, Steam Condensers, and Fans. All of these products are designed in-house. Prior to the use of the Macintosh computer two and a half years ago, all of these calculations were done using batch software on a mini or mainframe computer or were done by hand. The majority of the software that existed was written in FORTRAN.

Much of the software that is used today had their beginnings in the 60's and 70's, and have gone through many transitions. The ideal solution was to rewrite the software and make it easier to maintain, but the decision to discontinue the use of the mini and mainframe computers as soon as possible was a major factor in some of our development decisions. Our first two concerns were the user interface and the linking of that interface to a particular software package.

The User Interface:

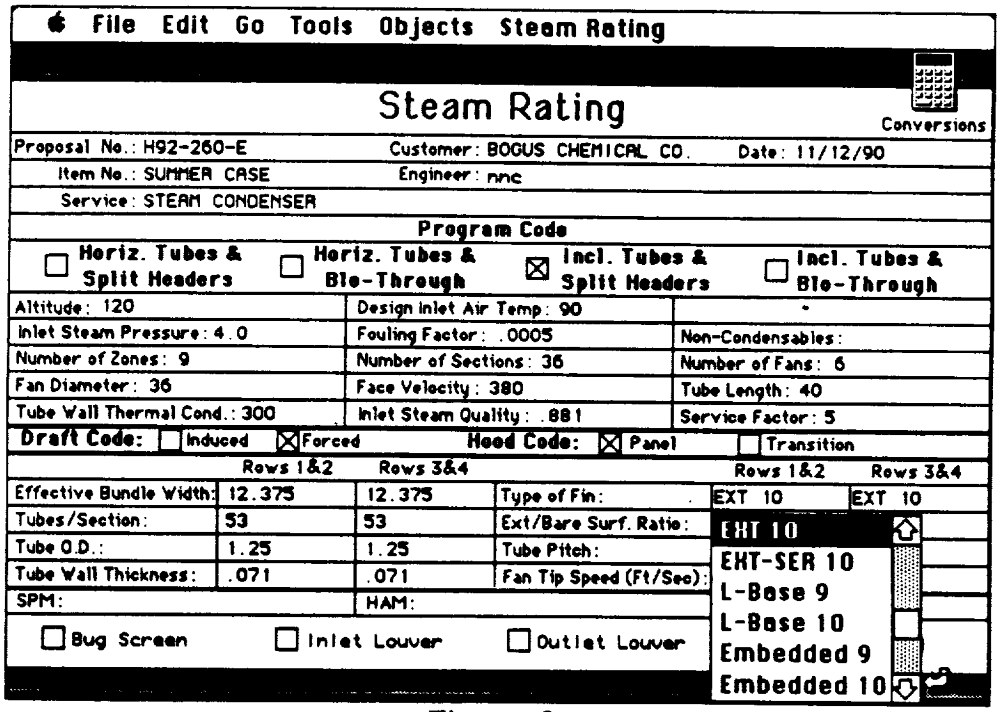
HyperCard was chosen as the development environment for the user interface. The ease and speed in which a particular interface could be created and altered was HyperCard's strongest point. Having chosen HyperCard, we set out on a user interface design. This seemed like an easy task, but it was soon discovered to be more difficult than had been anticipated. The engineers (as with most users) were set in their ways and initially were not open to a great deal of change.

For example, the three lines of text in figure 1 are a sample of the input that the engineers created prior to the use of HyperCard. Compare this to the screen depicted in figure 2 that is used today. Both figures reflect the same data.

```
BOGUS CHEMICAL CO.  H92-260-E       STEAM CONDENSER     SUMMER CASE   11/12/90nnc
11   900   120    4000       0    5     09 361 1      63600380400 300 0 088100     5     0
 1238   531250   710 3 19903123853    0    0 1238   531250   710 3 19903123853    00000
```

Figure 1.

When the development team came up with its initial design they assumed that the user would be comfortable with the point and click environment that the Macintosh provided. Feedback from the existing users showed a great desire to keep hand movement to a minimum, but the development team wanted to make the interface easy to use for experienced Macintosh users as well.



Steam Rating

Proposal No.: H92-260-E Customer: BOGUS CHEMICAL CO. Date: 11/12/90

Item No.: SUMMER CASE Engineer: nnc

Service: STEAM CONDENSER

Program Code

☐ Horiz. Tubes & Split Headers ☐ Horiz. Tubes & Blo-Through ☒ Incl. Tubes & Split Headers ☐ Incl. Tubes & Blo-Through

Altitude: 120	Design Inlet Air Temp: 90	•
Inlet Steam Pressure: 4.0	Fouling Factor: .0005	Non-Condensables:
Number of Zones: 9	Number of Sections: 36	Number of Fans: 6
Fan Diameter: 36	Face Velocity: 380	Tube Length: 40
Tube Wall Thermal Cond.: 300	Inlet Steam Quality: .881	Service Factor: 5

Draft Code: ☐ Induced ☒ Forced Hood Code: ☒ Panel ☐ Transition

	Rows 1&2	Rows 3&4		Rows 1&2	Rows 3&4
Effective Bundle Width:	12.375	12.375	Type of Fin:	EXT 10	EXT 10
Tubes/Section:	53	53	Ext/Bare Surf. Ratio:		
Tube O.D.:	1.25	1.25	Tube Pitch:		
Tube Wall Thickness:	.071	.071	Fan Tip Speed (Ft/Sec):		
SPM:		HAM:			

☐ Bug Screen ☐ Inlet Louver ☐ Outlet Louver

Figure 2.

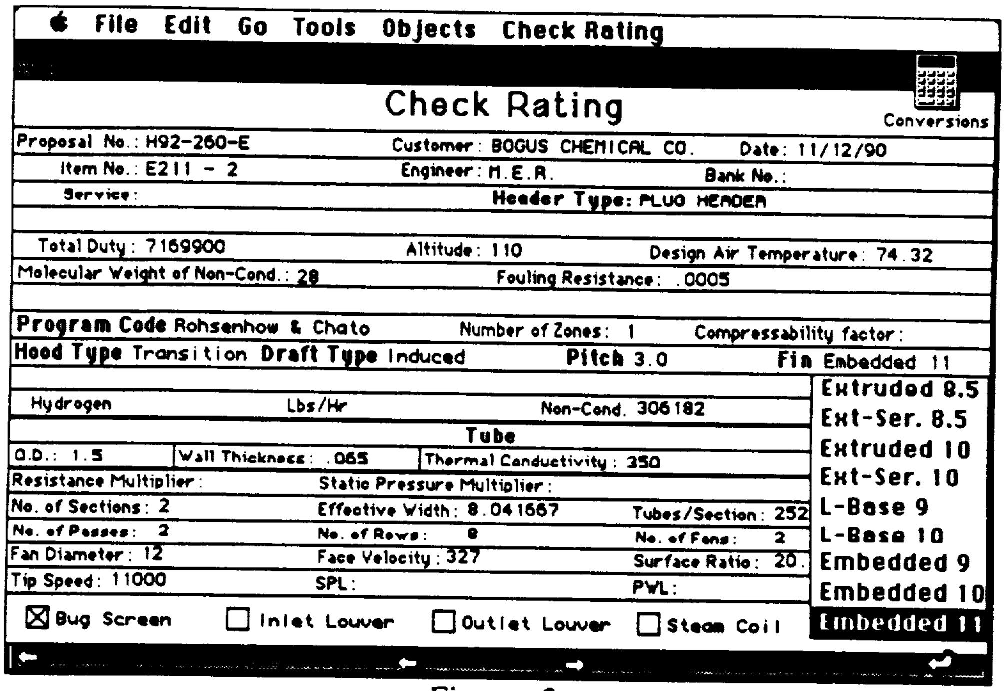
The engineers wanted to use the mouse as little as possible, the development team wanted to make the data entry as fool-proof as possible. Consequently the use of pop-up menus was made when there was a limited choice for input. This required the user to move from the keyboard, to the mouse and back again. The solution was to produce some code that allowed the user to "tab into" a menu, or a "tabbable pop-up menu". An example of this is the scrollable list seen in figure 2. The bulk of the code that did this was in an XCMD, (an XCMD is basically a subroutine written in some high-level language and then physically linked to the HyperCard stack).

The data for this card is entered by using the tab key to go from field to field and typing in the data and then typing return or tab to go to the next field. A field is indicated by a label to the left of the field, the label will contain a colon so as to distinguish it from the data entered. Some fields, such as the one labeled "Type of Fin" in figure 2, would present a pop-up menu or list. The user could then use the arrow keys to change the selection and press tab or return to enter the data, or press the escape key to leave the entry unchanged. The different groups of buttons on this card represent different flags that were set in the original input.

The icon labeled "Conversions" is a button that provided a link to another stack that was developed by one of the engineers. The "Conversions" stack allowed the engineers to perform unit conversions to and from a variety of units. A secondary benefit provided by this stack is that the engineers now all use a common set of conversion factors.

After getting the feedback about the "Steam Rating" stack we went on to develop a stack for another program called "Check Rating". Figure 3 shows the card that was created. Again we used the "tabbable pop-up menu", but also made less use of buttons on this stack. By using our pop-up menus instead of buttons, most of the input was kept at the keyboard, where the user wanted it, but the user still had the ability to use the mouse in the usual way. The fields that have a pop-up menu have boldfaced labels, figure 3 shows an example of the menu for "Fin".

The user interface may look like a trivial problem on the surface, but we found a large portion of our time devoted to issues such as those mentioned above. HyperCard proved to be flexible enough to allow us try a variety of solutions to a given user interface problem, and thus pick a solution that fit our needs the best.



Check Rating

Proposal No.: H92-260-E Customer: BOGUS CHEMICAL CO. Date: 11/12/90
Item No.: E211 - 2 Engineer: M.E.R. Bank No.:
Service: Header Type: PLUG HEADER

Total Duty: 7169900 Altitude: 110 Design Air Temperature: 74.32
Molecular Weight of Non-Cond.: 28 Fouling Resistance: .0005

Program Code Rohsenhow & Chato Number of Zones: 1 Compressability factor:
Hood Type Transition Draft Type Induced Pitch 3.0 Fin Embedded 11

Hydrogen Lbs/Hr Non-Cond. 306182

Tube
O.D.: 1.5 Wall Thickness: .065 Thermal Conductivity: 350
Resistance Multiplier: Static Pressure Multiplier:
No. of Sections: 2 Effective Width: 8.041667 Tubes/Section: 252
No. of Passes: 2 No. of Rows: 8 No. of Fans: 2
Fan Diameter: 12 Face Velocity: 327 Surface Ratio: 20.
Tip Speed: 11000 SPL: PWL:

[✓ Bug Screen] Inlet Louver Outlet Louver Steam Coil

Figure 3.

Linking the Interface:

Getting the user interface linked to the code that does all of the real work was another problem. We had a series of programs that originally ran in a batch mode that were written in FORTRAN. HyperCard had no direct way of communicating with this code. Normally an XCMD would be written to handle a series of calculations that would be difficult to implement in HyperTalk, but the time necessary to rewrite the code in C or Pascal was too involved.

We decided to use a third party tool that allowed us to convert FORTRAN code into an XCMD. This tool involved installing an XCMD into every stack that needed to access the FORTRAN code. The stack could then invoke this XCMD and pass information indicating which FORTRAN routine to run. The XCMD contained a runtime environment for the FORTRAN which allowed HyperCard to pass information to the FORTRAN code and the FORTRAN code to pass information back.

This solution worked well at first, but several problems occurred. The first problem involved the implementation of the third party tool. It was not always consistent in the way that it located the FORTRAN subroutine to be run. The initial design required the user to have a folder that contained the stack and a subsequent folder that contained the FORTRAN code, as shown in figure 4, but due to the way that this tool worked the FORTRAN code had to be placed in the System folder instead.

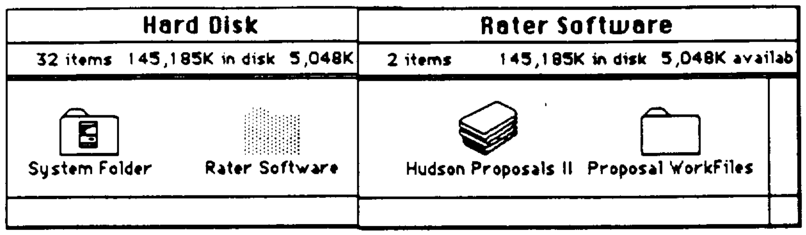


Figure 4.

The second problem with this solution was the time and memory overhead involved. The FORTRAN runtime package and subroutines required that at least 1.5 megabytes be allocated to the HyperCard program. This was acceptable, but not always desirable. The time overhead involved communication between HyperCard and the FORTRAN code. The third party XCMD situated itself between HyperCard and the FORTRAN and caused undesirable speed reduction in passing parameters back and forth. In fact we discovered that it was faster to write the parameters out to an external file and read them back in.

Several tests were conducted to solve each of these individual problems. The decision to make each piece of FORTRAN code a stand alone program seemed the best solution. This decision allowed us to revert to our original design of having the stack and a folder containing the FORTRAN programs in a directory of their own as shown in figure 4. This made software updates easier to manage since the user could copy the contents of one directory to his disk and not worry about putting different files in different locations.

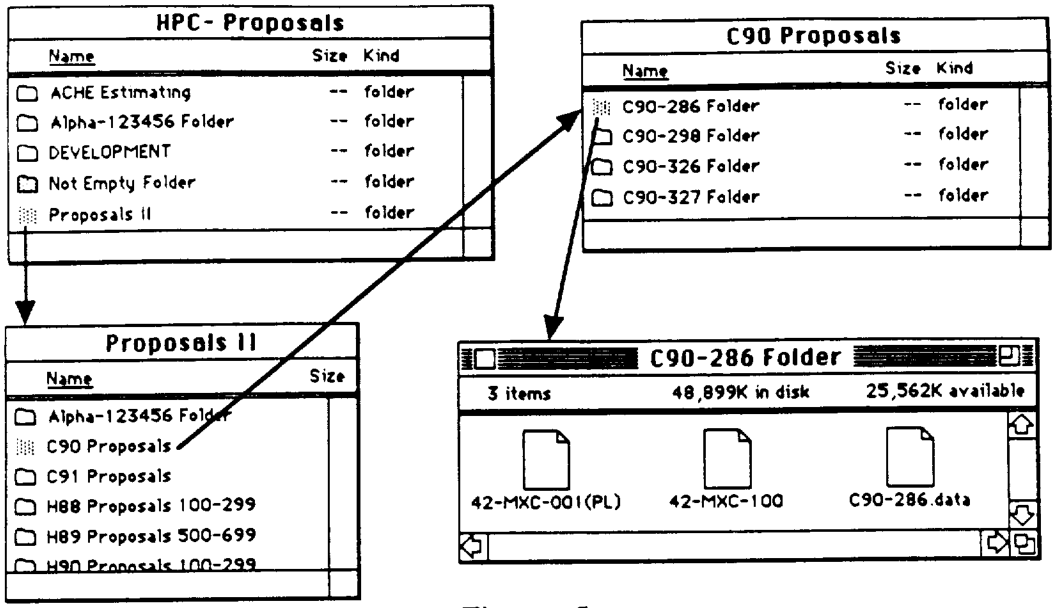
Our communication mechanism was simple. HyperCard would gather input from the user, write that data out to a file on disk, and then invoke the FORTRAN program. The FORTRAN program would read in the data, calculate the results, and write them out to another file, which would be read in by HyperCard and then reported to the user. All of this worked quickly enough to satisfy our users.

Other Concerns:
Another concern after getting the rudimentary portion of our stacks to work, was the need to store and fetch data between runs. We needed a database management system, but that area was still being studied and a complete solution had not been decided on, so a temporary solution was devised.

The easy solution was to save a copy of each stack after the input had been made, but there was a lot of overhead in the stacks. In fact the stacks themselves had been merged into a single stack which totaled more than 250 Kilobytes. Saving this much information was going to exhaust our storage rather quickly. The size of the data that actually needed to be stored was around 5 Kilobytes. So a scheme was devised that allowed us to store and retrieve the data in a text file. This technique required that the card design not change drastically. The most important restriction was that the deletion of fields or buttons was not allowed. We could add some if needed, but could not delete them and still be backwards compatible with our data format.

Using the Hierarchical File System of the Macintosh we achieved a pseudo-database that allowed our users to fetch related data. This worked by putting related files into the same folder, (see figure 5 for an example of this hierarchy). This technique lasted for about a month before two other problems became apparent.

The first problem involved the time needed to read and write the data files. Several solutions were tried, but the only one that achieved any appreciable time reduction was an XCMD. So we coded an XCMD to collect the data in memory and then dump it out to the disk. The drawback to this was that if we ever added new fields or buttons, then we had to recode the XCMD. This was acceptable.


HPC- Proposals

Name	Size	Kind
ACHE Estimating	--	folder
Alpha-123456 Folder	--	folder
DEVELOPMENT	--	folder
Not Empty Folder	--	folder
Proposals II	--	folder

C90 Proposals

Name	Size	Kind
C90-286 Folder	--	folder
C90-298 Folder	--	folder
C90-326 Folder	--	folder
C90-327 Folder	--	folder

Proposals II

Name	Size
Alpha-123456 Folder	
C90 Proposals	
C91 Proposals	
H88 Proposals 100-299	
H89 Proposals 500-699	
H90 Proposals 100-299	

C90-286 Folder

3 items 48,899K in disk 25,562K available

42-MXC-001(PL) 42-MXC-100 C90-286.data

Figure 5.

The second problem became two-fold. Users were saving data on each of their individual systems. This sometimes led to a duplication of information when more than one engineer worked on the same problem. The solution here was to modify the stack so that it could use a file server instead of the user's hard disk. This was accomplished without a significant amount of effort, but within 3 months we hit another barrier. The number of files created was getting rather large. Since this was a temporary solution to begin with, we simply added another level of hierarchy to the storage of the data.

Printing was a problem that was handled two ways. The first was for simple output, the solution was an XCMD that printed multi-page data in a single font. The second was for more complex output. We needed to create detailed forms and fill in the form with data from the HyperCard stack. For this we turned to a third party product called "Reports!". The initial version did the job, but was a bit difficult to implement, this product has been revised now and is well worth using.

Future Goals:

We have already started to implement an SQL database that HyperCard will access. This will allow us to link not only our engineering data, but our accounting, scheduling, and statistical data as well. Other plans include on-line help systems for each of the stacks, simple expert systems, and recoding some of our software into true XCMDs.

Evaluation:

HyperCard has its shortcomings, namely speed and lack of color. The speed issue can usually be worked around by use of a compiler or an XCMD. Color is a problem that will have to wait until Apple does something about it. Fortunately our needs do not necessitate the use of color. Card dimensions had caused us some small problems, but they too can be worked around.

HyperCard's biggest benefit is its ease of creating a user interface. Since it is easy to make modifications to the card design many different solutions to a particular problem can be evaluated. We have also been able to create a series of tools that all have the same "feel" to the user. This has helped keep the need for user training down significantly.

HyperCard has allowed us to accomplish more than we could have using conventional development methods. It has proven to be an excellent prototyping tool, and a more than adequate environment for the implementation of our software. Due to the success with this project, new projects are being implemented using HyperCard and the techniques described.

LIFE SCIENCES ON-LINE: A STUDY IN HYPERMEDIA APPLICATION

Life Sciences Project Division, NASA, Johnson Space Center

by

Linda A. Christman
Nam V. Hoang
David R. Proctor